A model to analyse the causality in synchronous real time systems

Albert Benveniste

IRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP105
78153 Le Chesnay Cedex
France
Tel (3) 954 90 20

Rapports de Recherche

N° 411

A MODEL TO ANALYSE THE CAUSALITY IN SYNCHRONOUS REAL TIME SYSTEMS

Albert BENVENISTE

Mai 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Publication Interne n° 252
Avril 1985
28 pages

A MODEL TO ANALYSE THE CAUSALITY IN SYNCHRONOUS REAL TIME SYSTEMS

Albert BENVENISTE
IRISA/INRIA
Campus de Beaulieu
F 35042 RENNES CEDEX
FRANCE

ABSTRACT: we present a model to handle the causality in synchronous real time systems. Synchronous real time systems communicate with the external world through a specified fixed set of interfaces. Among them are the input stimuli, which mainly govern the behaviour of the system. The present model takes into account that events as well as processes are indeed functions of these input stimuli; to study the corresponding objects, the notions of clock and causal process are introduced and studied. The present study served as one of the bases for the design of the language SIGNAL.

RESUME: nous présentons un modèle pour étudier la causalité dans les systèmes temps-réel synchrones. Ces systèmes communiquent avec le monde extérieur à travers un ensemble fixe et connu d'interfaces. Parmi ceux-ci figurent les stimuli d'entrée, qui pilotent le fonctionnement du système. Le modèle que nous présentons prend en compte le fait que processus et évènements sont en fait des fonctions de ces stimuli d'entrée; afin d'étudier les objets correspondants, on introduit les notions d'horloge et de processus causal. Ce modèle a constitué l'une des bases de réflexion pour la conception du langage SIGNAL.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

1. INTRODUCTION.

According to Young (1982), a real time system is "any information processing activity or system which has to respond to externally generated input stimuli within a finite and specifiable delay". A further feature of *synchronous* real time systems is that to every event can be assigned a time at which this event occurs. A consequence is that, to avoid any non determinism, the set of the input stimuli has to be fixed and specified in advance. Since our aim is to study the notion of event in real time systems, we shall consider events as *functions of the input stimuli;* this will allow us to take into account the causality constraints in a natural framework. Note that this is exactly the point of view of Le Guernic & al.(1985); a suited formalism is presented in a preliminary form in Caspi & Halbwachs (1984), and we shall in fact follow the same approach in a more systematic way. We should say that the ideas underlying the present approach go back to the classical point of view of the probability and ergodic theories to handle the time (Dellacherie & Meyer (1976), Benveniste (1974)).

1.1. What is the "time" in real time synchronous systems ?

While classical (i.e. non asynchronous) real time languages do implicitely or explicitely refer to some external and universal time reference, the notion of "time" is completely different in synchronous real time systems. To be more explicit, synchronous real time systems differ from asynchronous ones in the following aspects:

(*) concerning the internal mechanisms of the system: every action is instantaneous , i.e. has a zero duration.

(*) concerning the communications with the external world: the set of the possible input stimuli is fixed and known in advance, and input lists are specified

through both 1/ the values their carry, 2/ a total ordering of the "instants" at which these values are available at the input ports.

Of course, the specification 2/ above is the fundamental feature which characterizes the way synchronous real time systems communicate with the external world, compared to asynchronous ones. Let us illustrate further this point on a simple example.

Consider a real time system, which has two inputs:

(i) a data input carrying an ordered file of data named x,

(ii) an interrupt input port named s.

Then, the specification of an input history according to the synchronous point of view must be of the form

$$x_1,(x_2,s_2),x_3,s_4,etc...$$

i.e. both the values and their global ordering must be specified; the integer index t= 1, 2,... is used for this purpose. And *this index "t" has to be considered as the proper notion of time in synchronous systems*. In other words, the essentially nondeterministic character of the communications with the external world in real time systems is concentrated here inside some (ignored) external mechanism which *decides* this global ordering. Hence, the advantage of the synchronous point of view is that the nondeterminism of the external communications is strictly concentrated in this external mechanism, and is thus by no means propagated inside the body of the system itself. For this reason, G. Berry oftens refers to synchronous real time systems as *reactive systems*, i.e. systems which react instantaneoulsly to external input stimuli in a functional way.

1.2. What is the proper notion of process in reactive systems?

It is usual to consider that processes that have the same external behaviour are indistinguishable; usual descriptions of external behaviours in (asynchronous) real time systems are, for example, the pairs {traces, failure set} used in CSP (Brookes & al. (1984)). In reactive systems, however, the notion of external behaviour can be defined in a more rich way, thanks to the synchrony assumption.

Consider a real time system, and denote by

$$X_t = (x_t(1),\ldots,x_t(n)) \qquad (1)$$

the set of its input stimuli, where the index "t" refers to the time index. Denote by

$$X_{[1,t]} = (X_1, \ldots, X_t) \qquad (2)$$

the *initial segment* up to time t. Then, a natural definition of the external behaviour of a process is the set of its *traces* defined informally as *the collection of the inputs and outputs up to time* t, where t ranges the time index set. This is a refinement of the usual definition of traces, which takes into account the synchronous character of reactive systems; on the other hand, it is known that traces entirely characterize deterministic processes, see Brookes & al.(1984). As a consequence, the basic notion to study reactive systems is the set of the functions which map initial segments of the inputs to corresponding initial segments of the outputs, a point of view close to Kahn (1974); as usually, fixed point arguments can be used to define a global input-output map, the restrictions of which are consistent with the previously given collection. These maps will be referred to as *processes*. Since our model intend to support the language SIGNAL, which is not of imperative type, but rather of functional type, we need to be able to characterize the processes P which are correct in the following sense: can the output of P up to time t effectively be defined with the only knowledge of the

input up to time t? Such correct processes will be referred to as *causal processes*.

1.3. What is the proper notion of event for reactive systems?

Reactive systems are closed worlds, since they do communicate with the external world through a fixed set of input stimuli. As a consequence, events can be caused only by the monitoring of the input stimuli, or of any of the outputs of some process. In this way, events appear naturally as *functions of the input stimuli*. Such functions will be referred to as *clocks*. Suitable time correctness conditions can be also introduced for clocks, and will be defined subsequently.

To summarize, the questions we have to answer are the following:

(i) How to guaranty that a data requested at time "t" can be effectively produced with the initial segment $X_{[1,t]}$? An infinite file indexed with the time of such data will be referred to as a *causal process*.

(ii) How to characterize events that are produced by the monitoring of causal processes ? Such sequences of events will be referred to as *clocks*. Clocks can be used to generate *time changes*, and new causal processes and clocks can be generated subjected to this new timing, and so on.

(iii) How to guaranty that this procedure will preserve the causality with respect to the original input stimuli ?

(iv) What are the convenient primitive operations on the time to be able to build any new time reference from the original one ?

2. BASIC NOTIONS: HISTORIES, CAUSAL PROCESSES, CLOCKS.

2.1. Histories.

In the sequel, the symbols **N** and N_0 denote respectively the ordered sets $\{1,2,\ldots,\infty\}$ and $\{0,1,\ldots,\infty\}$.

DEFINITION 1: By a *history*, we have in mind an object

$$\left\{\Omega,(\Pi_t)_{t\in N_0}\right\}$$

where

* Ω is a set.

* for every t, Π_t is a partition of the set Ω the members of which are referred to as *atoms*.

* for s<t , Π_t is *finer* than Π_s , denoted by

$$\Pi_s < \Pi_t,$$

i.e. every atom of Π_s is a union of atoms of Π_t.

The terminology is borrowed from Kahn(1974), and is in accordance with the usual notions of history, as the following example will show. When there is no ambiguity, we shall simply denote (Π_t) instead of $(\Pi_t)_{t\in N_0}$.

A basic example: the canonical history associated to a synchronous real time system.

Consider again a system whose input stimuli is denoted as in (1). Set

$$\Omega = \left\{N \to \mathbf{X}\right\} \tag{3}$$

where **X** denotes the set in which X_t takes its values, and denotes by ω the elements of Ω: every ω represents thus a possible trajectory $X_1,X_2,\ldots$ of the input X.

Then, Ω is endowed with the following equivalence relation

$$\omega \sim_t \omega' \text{ iff } \omega(s)=\omega'(s) \text{ for every } s \leq t \quad (4)$$

The partition Π_t is then defined as follows: ω and ω' belong to the same atom of Π_t iff $\omega \sim_t \omega'$. By convention,

$$\Pi_0 = \{ \phi , \Omega \}$$

represents the information available in the initial conditions (constants,...), which do not depend upon the values of the inputs. Then, $\{\Omega, (\Pi_t)\}$ is the canonical history associated to the input X; note that only the input is relevant in this definition. The partition Π_t is interpreted as *the information available at time t*, i.e. any data available at time t in the system is nothing but a function defined on the set Ω, which is constant on the atoms of the partition Π_t, i.e. a function of the initial segment $X_{[1,t]}$.

This formalism will be very easy to use, and covers without difficulty the case of several inputs with different data rates, provided that, as we mentioned before, the interleaving of these input streams be a part of their specification.

2.2. Clocks.

Our notion of clock is apparently different from, but in fact closely related to the notion of *event* used in Caspi & Halbwachs (1982,1985); connections also do exist with the work of Cardelli(1982). The main contribution is that we consider here clocks as functions, which map the input values ω into the set of the events (in the sense of Caspi & Halbwachs), and furthermore satisfy again appropriate causality constraints.

DEFINITON 2: Given an history $\{\Omega,(\Pi_t)\}$, a $\{\Omega,(\Pi_t)\}$-clock H (or, simply a *clock* when there is no ambiguity), is a time-indexed family of functions

$$H_t : \Omega \to \Xi \quad (5)$$

(where Ξ is a denumerable, totally ordered set containing N_0, and isomorphic to

N_0), satisfying the following properties

$$H_\infty(\omega) = 0 \text{ for every } \omega \tag{6.i}$$

$$s<t \text{ and } H_s(\omega)<\infty \text{ imply } H_s(\omega)<H_t(\omega) \tag{6.ii}$$

$$s<H_t(\omega)<s+1 \text{ and } \omega' \sim_s \omega \text{ imply } H_t(\omega')=H_t(\omega) \tag{6.iii}$$

The last condition is a causality condition: it expresses the fact that, to know if the t-th occurrence of an event H(ω) takes time strictly before s+1, it is sufficient to observe the initial segment up to time s. Finally, since our aim is to avoid any nondeterminism, any multiple occurrence of an event must be time ordered; hence, the use of oversets of N_0 is necessary, since we want to handle multiple events. The relevance of the time correctness constraint (6) will be subsequently illustrated in the forhtcoming examples and counterexamples.

Counters.

Counters are extensively used by Caspi & Halbwachs (1982, 1985) to monitor events in real time systems. Counters are associated to clocks as follows. Given a $\{\Omega,(\Pi_t)\}$-clock H, the formula

$$\mu_t^H(\omega) = \# \left\{ s \in N_0 : H_s(\omega) < t+1 \right\} \tag{7}$$

defines *the counter associated to H*. Note that it is generally not possible to recover H from its counter μ^H, unless $\Xi=N_0$, since counters do not provide any information about the ordering of the events which occur between t and t+1; this would cause difficulties when time changes are of interest, as we shall see later. Thanks to (6.iii), counters enjoy the following property:

LEMMA 1:

$$\omega' \sim_t \omega \text{ implies } \mu_t^H(\omega') = \mu_t^H(\omega)$$

The proof is easy, and is left to the reader. This lemma expresses that the counter associated to a clock is a causal process in the sense of the definition 4

below.

Examples and counterexamples.

1) Any strictly increasing function H: $N_o \to N_o$ which do not depend upon the input stimuli ω is a clock, since the causality constraint (6.iii) is trivially satisfied. The events generated by such clocks are known prior to the observation of the input stimuli, i.e. before the running of the program.

2) Take $\Xi = N_o^2$ endowed with the lexicographic order, defined by

$$(m,p)<(n,q) \text{ iff } [m<n] \text{ or } [m=n \text{ and } p<q]$$

and set

$$H_t = (s,k) \text{ if } t=sp+k \text{ with } 0\leq k<p$$

This is the simplest case of *oversampling*, where the new sampling rate do not depend upon the input stimuli. This procedure will be properly generalized later.

3) Consider a process X of type T, such that

$$\omega' \sim_t \omega \text{ implies } X_t(\omega') = X_t(\omega) \tag{8}$$

and let A be a subset of T. Define

$$\begin{aligned} H_0(\omega) &= 0 \\ H_1(\omega) &= \min\left\{s>0 : X_s(\omega)\in A\right\} \\ H_{t+1}(\omega) &= \min\left\{s>H_t(\omega) : X_s(\omega)\in A\right\} \end{aligned} \tag{9}$$

where, by convention, $\min(\phi)=\infty$. Then, H is a clock, for it satisfies the time-correctness constraint (6.iii) since X is a (causal) process.

4) Here follow two counterexamples. First, modify (9) as follows

$$\begin{aligned} H_1 &= \min\left\{s : X_{s+u}\in A\right\} \\ H_{t+1} &= \min\left\{s>H_t : X_{s+u}\in A\right\} \end{aligned}$$

where u>0, and ω was deleted for convenience. Then, H is clearely a $\{\Omega,(\Pi_{t+u})\}$-

clock, but not a $\{\Omega,(\Pi_t)\}$-clock, since the time-correctness constraint is violated. More subtle is the following example: let $k_1,k_2,\ldots$ be an increasing sequence of integers, define

$$H_t = \max\left\{s<k_t : X_s \in A\right\},\ \max(\phi)=0$$

Then, H is not a clock, since the time-correctness constraint is violated.

History associated to a clock.

This notion will be a first step towards the technique of time change. The idea behing time changes is the following: suppose you have a clock, and you are interested in an infinite ordered file of data which are available at each occurrence of this clock. Then, you would probably like to forget the original time reference, and prefer to work with the above mentioned clock *as it were the time reference*. Our aim is to justify such a procedure. A first step in this direction is the following definition:

DEFINITION 3: Let H be a $\{\Omega,(\Pi_t)\}$-clock. Define

$$\omega \sim_{H_t} \omega' \text{ iff } s\le H_t(\omega)<s+1 \text{ and } \omega\sim_s\omega' \qquad (10)$$

This is an equivalence relation; (10) defines $\{\Omega,(\Pi_{H_t})\}$ as *the history associated to the clock H.*

The fact that (10) defines an equivalence relation is due to the property (6.iii) of time-correctness of the clock H. This history summarizes the flow of information corresponding to every new occurrence of H, a fundamental notion when the study of time changes is of interest.

2.3. Causal processes.

Consider an history $\{\Omega,(\Pi_t)\}$. For every $\omega\in\Omega$, we shall denote by

$$\omega/t \tag{11}$$

the atom of Π_t which contains ω; referring to the canonical history of a system, ω/t is nothing but the initial segment of length t of ω. Then, the set of the initial segments of Ω is endowed with the following partial order

$$\omega/t_1 \leq \omega/t_2 \tag{12}$$

iff, $t_1\leq t_2$, and $\omega/t_1 \supset \omega/t_2$. Then, every chain $\omega/1 \leq \omega/2 \leq ...$ has a unique maximal element, which is nothing but the atom $\cap_t \omega/t$. This upper limit defines a unique point of Ω when the atoms of $\bigcup_{t=0}^{\infty}\Pi_t$ are the points of Ω, which is for example the case for the canonical history of a system. Then it is reasonable to define a causal process X as a time indexed set of functions of ω such that $X_t(\omega)$ depends only on the initial segment ω/t. The following definition generalizes this idea.

DEFINITION 4: Let H be a $\{\Omega,(\Pi_t)\}$-clock. By a $\{\Omega,(\Pi_t),H\}$-process (for short instead of *causal process*), we have in mind a time-indexed family $X = (X_t)$ of functions

$$X_t\colon \Omega \to T$$

(where T is a set defining the type of the process), satisfying the following *time-correctness* condition:

$$\omega' \sim_{H_t} \omega \text{ implies } X_t(\omega')=X_t(\omega) \tag{13}$$

The definition 2 expresses the fact that $X_t(\omega)$ depends only on the initial segment $\omega/H_t(\omega)$; this initial segment is well defined, since H is a clock. In other words, the value X_t has to be considered as available at time H_t.

3. THE CLOCK ALGEBRA.

The aim of this chapter is to study the set of all the $\{\Omega,(\Pi_t)\}$-clocks, we shall simply refer to as clocks, since no confusion can occur throughout this chapter. We shall introduce two primitive operations on this set, the *filtering*, and the *multiplexing*. And we shall prove that these primitives allows us to buit in finitely many steps any (reasonable!) clock.

3.1. A partial order on the set of the clocks.

It is defined as follows: given two clocks H and K, we say

$$K \subset H \text{ iff } \bigcup_{t=0}^{\infty}\left\{K_t(\omega)\right\} \subset \bigcup_{t=0}^{\infty}\left\{H_t(\omega)\right\} \tag{14}$$

for every ω. In other words, K⊂H means that the set of the occurrences of K is contained in the set of the occurrences of H *whatever the input ω is*, i.e. $\subset$ is a partial order on functions. This order is different from the order introduced in Caspi & Halbwachs (1982), the aim of which is the study of precedence of events; this order is however introduced in Caspi & Halbwachs (1985).

3.2. The filtering.

Every process of Boolean type will be said to be a *test*. In the sequel, we shall sometimes identify the boolean values "true" and "false" respectively with the integer values "1" and "0". Let H be a clock, and T a $\{\Omega,(\Pi_t),H\}$-test. Then, a new clock K is obtained by filtering H by T as follows:

$$K_t(\omega) = \max\left\{H_s(\omega) : \sum_{u=1}^{s} T_u(\omega) \leq t\right\} \tag{15}$$

To refer to the filtering operation, we shall use the notation

$$K = H\downarrow T \tag{16}$$

In other words, *$H\downarrow T$ extracts from H the instants H_s where T_s is true*. It is easy to verify that H↓T is still a clock. Conversely, the following result is true

LEMMA 2: If H'⊂H, then H' = H↓T, where the {Ω,(Π$_t$),H}-test T is given by

$$T_s(\omega) \quad = \text{true if } H_s(\omega) \in H'_s(\omega), \quad = 0 \text{ otherwise.} \tag{17}$$

In the formula (17), we used the notation

$$H_s(\omega) \quad \text{for short instead of} \quad \bigcup_t \left\{ H_t(\omega) \right\} \tag{18}$$

PROOF: elementary, left to the reader.

The filtering is the basic operation to build the set of all the clocks which are dominated by a single one. Note that this is the only operation, existing real time languages provide to handle the time; see for example the synchronous real time language ESTEREL (Berry & al.(1983), Berry & Cosserat (1984)), and the preliminary version of the language SIGNAL which is presented in Le Guernic & al.(1985); the usual instruction implementing the filtering is of course the IF statement. However, when the oversampling of data rates is of interest, which is actually very often the case, no tool is provided to the programmer to handle properly the corresponding time index. Our purpose is to investigate theoretically the difficulties behind the notion of oversampling. The corresponding operator will be the multiplexing, for short instead of *time-multiplexing*.

3.3. The multiplexing.

Let H be a clock taking values in the overset Ξ of N_0 (see the definition 2), and let C be a causal {Ω,(Π$_t$),H}-process of integer type, such that $C_t > 0$ for t finite, and $C_\infty = 0$. Set

$$\Xi' := \Xi \times N \tag{19}$$

endowed with the *lexicographic order*, defined by

$$(t,k) < (t',k') \text{ iff } (t < t') \text{ or } (t = t' \text{ and } k < k') \tag{20}$$

where, by convention, $(\infty,k) = \infty$ for every k. Then, the *multiplexing of the clock H by the ceeling (†) function C* is denoted and defined as follows:

(†) this name, originated from architecture theory, is justified by the fig.(1), where the depicted object looks somewhat like Manhattan.

$$K = H\uparrow C \tag{21}$$

$K_t(\omega) =$

$= (H_s(\omega), u)$

if $\exists s$: $0<u\leq C_s(\omega)$ and $t = C_1(\omega)+\ldots+C_{s-1}(\omega)+u$

$= \infty$ otherwise

The fig.(1) depicts this procedure; the ↑ arrows mean an increase by 1 of the second component of the clock $H\uparrow C$, whereas the ↓ replaces at the same time H_s by H_{s+1}, and $u = C_s$ by u=0.

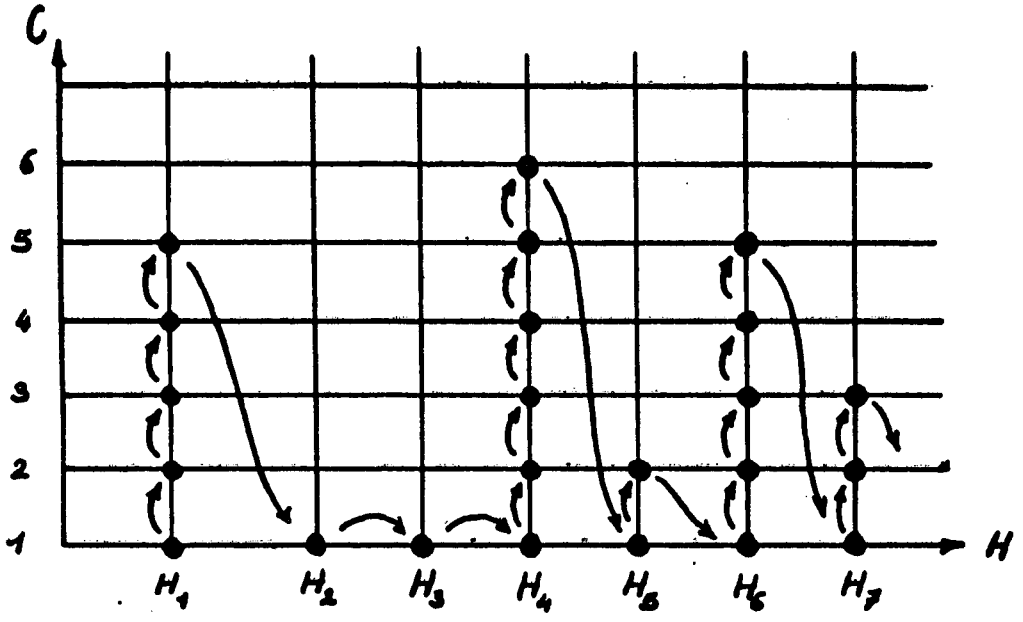

fig. (1): construction of $H\uparrow C$

The following result holds:

THEOREM 1: K=$H\uparrow C$ is a $\{\Omega,(\Pi_t)\}$-clock.

PROOF: We have only to verify the time correctness property (6.iii). Using (20) and (21), the inequality

$$v\leq K_t(\omega) <v+1$$

is equivalent to

$$v\leq H_s(\omega) <v+1 \tag{22}$$

Choose ω' such that $\omega'\sim_v\omega$. Then, (23) and the fact that C is a process of clock H

imply that

$$H_s(\omega') = H_s(\omega) \tag{23.i}$$

$$C_k(\omega') = C_k(\omega) \text{ for every } k \leq s \tag{23.ii}$$

Finally, (21) and (23.i,iii) imply the desired property, namely

$$K_t(\omega') = K_t(\omega) \tag{24}$$

This finishes the proof.

The following theorem is the first fundamental result. It expresses the fact that allmost all $\{\Omega,(\Pi_t)\}$-clock can be obtained by applying finitely many times the operations *filtering* and *multiplexing*. Hence, these operations can be considered as primitives to build any clock from an original one. Corresponding primitives instructions are provided in the language SIGNAL, see Le Guernic & al.(1984) and (1985).

THEOREM 2: Let H be a $\{\Omega,(\Pi_t)\}$-clock taking values in the set

$$\Xi := N_0 \times N^{k-1}, \quad 0<k<\infty \tag{25}$$

endowed with the lexicographic order. Then H can be decomposed as follows

$$H(0) = Id, \tag{26.i}$$

$$H(1) := H(0)\downarrow T(1) \tag{26.ii}$$

$$\text{for } m>0, \quad H(m) := H(m-1) \uparrow C(m) \downarrow T(m), \tag{26.iii}$$

$$H := H(k) \tag{26.iv}$$

where, for m=1,...,k, T(m) (resp. C(m)) are suitable tests (resp. ceeling functions), and Id denotes the clock "identity" defined by

$$Id_t(\omega)=t \tag{27}$$

Furthermore, among all the possible decompositions (26), there is a *minimal* one, we shall denote by H*(0),...,H*(k), such that

$$H^*(m) \subset H(m) \tag{28}$$

for any decomposition associated to H through (26).

COMMENT: Ξ's of the form N^k are not the most general, so that the theorem do not cover all possible clock. However, N^N endowed with the lexicographic order is nothing but the (non denumerable) set of the denumerable ordinals, which is of no practical interest for us. Anyway, Ξ's of the form N^k are the only for which a construction of clocks in finite steps is possible.

PROOF: by induction over the index m. For m=1, the result is a direct consequence of the lemma 2. For m>1, we can decompose H(m) as follows

$$H(m)_t(\omega) = \left[h_t(\omega), u_t(\omega)\right] \in N^{m-1}\times N \tag{29}$$

The values of $h_t(\omega)$ are non decreasing in N^{m-1}. Group them as follows

$$\begin{aligned} h_1(\omega)=\ldots=h_{D_1(\omega)}(\omega) &:= H(m\text{-}1)_1(\omega) \\ < h_{D_1(\omega)+1}(\omega)=\ldots=h_{D_1(\omega)+D_2(\omega)}(\omega) &:= H(m\text{-}1)_2(\omega) \\ <\ldots \text{ and so forth} & \end{aligned} \tag{30}$$

Our first task is to verify that the so defined H(m-1) is indeed a clock; as usually, the key property to verify is the time correctness property (6.iii). According to the definition of the lexicographic order, we have

$$s\le H(m)_{t'}(\omega)<s+1 \text{ iff } s\le H(m\text{-}1)_t(\omega)<s+1 \tag{31}$$

where t and t' are related through

$$H(m)_{t'}(\omega) = \left[H(m\text{-}1)_t(\omega), u_t(\omega)\right] \tag{32}$$

Using (32), we get

$$\begin{aligned} &\omega' \sim_s \omega \text{ implies} \\ &H(m)_v(\omega') = H(m)_v(\omega) \text{ for } v\le t' \end{aligned}$$

which implies

$$H(m-1)_i(\omega') = H(m-1)_i(\omega) \text{ for } i\le t,$$

so that H(m-1) is a clock. On the other hand, taking

$$C(m)_t(\omega) = \max\left\{u\ge 0: \exists\, t' : H(m)_{t'}(\omega) = (H(m-1)_t(\omega), u)\right\} \tag{33}$$

we get (26.ii) with T(m) given by the lemma 1. Clearely, the decomposition we

have built is precisely the minimal decomposition of the clock H. This finishes the proof.

WARNING: The decomposition (26) is by no means unique. For example, take any ceeling function C' associated to *Id*, and choose a Id↑C'-test T' which deletes from Id↑C' some entire segments of the form $[(t,1),\ldots,(t,C'_t)]$. Then, clearely, the corresponding "t" indices could be deleted directly from the original clock Id, before the ceeling function C' operates.

A FUNDAMENTAL REMARK: existing real time synchronous languages explicitely assume that every clock is defined in terms of the *finest* one. The usual underlying argument is that, when a new faster clock has to be introduced in the program, then all the events have to be reexpressed with respect to this faster time reference. This argument is incorrect, as shown by the theorems 1 and 2 above: *the master clock* (when it exists) *is not necessarily the fastest one, but is rather the clock from which every other one can be derived.* For example, if C is an input dependent ceeling function, the clock Id↑C (where Id denotes the identity clock: $Id_t = t$) is faster than Id, but uses C to be built, so that Id↑C is derived from Id, while the converse is not possible.

3.4. Synchronizable clocks.

DEFINITION 5: Two clocks H and H' are said to be *synchronizable* if their associated counters μ^H and $\mu^{H'}$ are equal.

Synchronizable clocks are not distinguished in the framework of Caspi & Halbwachs, since these authors work only with counters. However, we preferred to distinguish them (when they are not equal!), since the instruction set of the language SIGNAL will allow the programmer to construct objects as in (26),

whereas the condition of synchronizability is generally very difficult to check. The name of "synchronizable" is justified by the following result:

THEOREM 3: If H and H' are synchronizable, then their associated histories $\{\Omega,(\Pi_{H_t})\}$ and $\{\Omega,(\Pi_{H'_t})\}$ are equal. In other words, two synchronizable clocks define the same flow of information.

PROOF: Denote respectively by (H(m)) and (H'(m)) the minimal decompositions (26) associated to H and H'. The clocks H and H' are synchronizable, if and only if the following property holds:

$$s \leq H_t(\omega) < s+1 \text{ iff } s \leq H'_t(\omega) < s+1 \tag{34}$$

which proves the result, thanks to (10).

It is clear that the matching of synchronizable clocks is nothing but the reordering of the set of the events which are not anterior to t, but anterior to t+1. Since our ultimate goal is the design of a synchronous language, we must provide to the programmer tools to specify such an ordering (to avoid any non-determinism in the language); the theorem 3 indicates the degree of freedom which is available to the programmer.

4. TIME CHANGES.

As it were mentioned in Le Guernic & al.(1985), time changes is a key issue when modularity is required in the language SIGNAL. Roughly speaking, the problem we want to solve is the following. Suppose you have written a program whose input stimuli are summarized in the flow $X=(X_s)$, and you want now to consider this flow X as produced by some other program by identifying X with some output, say Y, of this former program. When Y is a H-causal process, where the clock H differs from the identity, such a connection results in a change in the

time reference for the program with input X. In such a case, are the notions of causal process and clock conveniently embedded in the so obtained new time reference? Before to investigate the problem in a general setting, we shall further develop this first example.

4.1. Cascading systems.

Consider an history $\{\Omega,(\Pi_t)\}$; let H be a clock on this history. Let X be a $\{\Omega,(\Pi_t),H\}$-causal process defined on the same history. For example, if X' is a $\{\Omega,(\Pi_t)\}$-process of real type, we can obtain such an X by delivering the current value of X' only when it exceeds some threshold. Then, X is an output of the considered system, but X can on the other hand also be used as an input to some other system. To study this cascaded connection, let us introduce the canonical history $\{W,(\Gamma_n)\}$ associated to X (see (2.1)). The mapping

$$\Phi:\Omega \to W, \quad \omega \to w=\left\{X'_1(\omega),X'_2(\omega),\ldots\right\}$$

defines a connection from Ω to W. Here, an abstract model of modular programming is the ability to define processes and clocks on W, and to embed them using Φ back to processes and clocks on Ω. The purpose of this example was to illustrate the theoretical development we shall give now.

4.2. Connections.

DEFINITION 6: Consider two histories $\{\Omega,(\Pi_t)\}$ and $\{\Omega',(\Pi'_t)\}$, a $\{\Omega,(\Pi_t)\}$-clock H, and a subset $W \subset \Omega\times\Omega'$. We define the *connection of* $\{\Omega,(\Pi_t)\}$ *to* $\{\Omega',(\Pi'_t)\}$ *through* (W,H) as being the history

$$\{ W, (\Gamma_t) \} := \left[(W,H): \{ \Omega, (\Pi_t) \} \to \{ \Omega', (\Pi'_t) \}\right] \tag{35}$$

where

$$\Gamma_t := \left[\Pi_t\times\Pi'_{\mu_t^H}\right]_{|W} \tag{36}$$

where μ^H denotes as usually the counter associated to H (see (7)), and "$\bullet_{|W}$" denotes the restriction of $\bullet$ to W.

There is no ambiguity in this definition, since $w_1=(\omega_1,\omega'_1)$ and $w_2=(\omega_2,\omega'_2)$ belong to the same atom of $\Pi_t \times \Pi'_{\mu^H_t}$ if and only if, first,

$$\omega_1 \sim_t \omega_2 \tag{37}$$

which turns out to imply

$$\mu^H_t(\omega_1) = \mu^H_t(\omega_2)$$

(since counters are causal processes by virtue of the lemma 1), and second,

$$\omega'_1 \sim_{\mu^H_t(\omega_1)} \omega'_2 \tag{38}$$

Note that the time reference of the first component ω of w is the time "t" of Π_t, whereas the time reference of the second component ω' of w has been changed to the clock H defined over Ω. Hence, the connection is not a symmetric operation, for it gives a privilege to the time reference of one of the two histories.

4.3. Examples of connections.

4.3.1. Cascading histories.

This example is a formal rewriting of the previous one. Consider a map

$$\Phi : \Omega \to \Omega'$$

satisfying to the following causality condition:

$$\omega_1 \sim_t \omega_2 \text{ implies } \Phi(\omega_1) \sim_{\mu^H_t} \Phi(\omega_2) \tag{39}$$

This map is the convenient way to describe the connection between the two programs we have informally described before. Then, set

$$W := \text{graph of } \Phi \tag{40}$$

The structure of this connection is quite simple, since the so obtained history $\{W,(\Gamma_t)\}$ is indeed *isomorphic* to the history $(\Omega,(\Pi_t)\}$ via the map

$$\omega \rightarrow (\omega, \Phi(\omega))$$

We have thus described a simple input-to-output connection. The next example is more involved.

4.3.2. Closing loops.

Roughly speaking, this case corresponds to the instruction "@" of the language SIGNAL, which connects output ports of a SIGNAL-process to its input ports which have the same name. A formal description of loop closures is as follows. Take $\{\Omega',(\Pi'_t)\} = \{\Omega,(\Pi_t)\}$, H=Id, and let

$$W \subset \Delta_{\Omega\times\Omega} \tag{41}$$

where $\Delta_{\Omega\times\Omega}$ denotes the diagonal of the set $\Omega\times\Omega$. The fact that W is contained in this diagonal indicates that the inputs of W correspond to *identical* components, so that a single program is indeed involved in this connection. The fact that W is a *subset* of this diagonal expresses that constraints are set on the inputs by this connection, thus reducing the degree of freedom at those inputs. For example, if ω=(x,y) and z=z(x,y) is some causal process, then a loop closure is for example defined by ω=(x,f(z)) for some suitable function f.

4.4. Time correctness of connections.

4.4.1. Clock transfers.

Let

$$\{ W, (\Gamma_t) \} := \left[(W,H): \{ \Omega, (\Pi_t) \} \rightarrow \{ \Omega', (\Pi'_t) \} \right] \tag{42}$$

be a connection.

If K is a $\{\Omega,(\Pi_t)\}$-clock, the connection creates a $\{W,(\Gamma_t)\}$-clock by simply taking

$$L_t(\omega,\omega') = K_t(\omega) \tag{43}$$

The transfer of clocks defined on the second component is more complex, since

it involves a time change. Consider thus a $\{\Omega',(\Pi'_t)\}$-clock K'. Decompose it according to the formulas (26) of the theorem 2:

$$K' = \left[\left[\cdots\left[Id\downarrow T(1)\right]\cdots\right]\uparrow C(k)\downarrow T(k)\right] \tag{44}$$

Then, set

$$L(\omega,\omega') = \left[\left[\cdots\left[H(\omega)\downarrow T(1)(\omega')\right]\cdots\right]\uparrow C(k)(\omega')\downarrow T(k)(\omega')\right] \tag{45}$$

The right handside of (45) do not depend upon the particular decomposition of K', so that L is well defined. To prove that L is a W-clock, it is equivalent to prove that the counter μ^L associated to L via (7) is a W-causal process. This is a direct consequence of the formula

$$\mu^L_t(w) = \mu^L_t(\omega,\omega') = \mu^K_{\mu^H_t(\omega)}(\omega') \tag{46}$$

and of (37) and (38). To summarize, we proved that

THEOREM 4: connections do preserve the time correctness of clocks.

4.4.2. Transfer of causal processes.

Consider again a connection (42). If X is a $\{\Omega,(\Pi_t),K\}$-causal process, then the formula

$$\begin{aligned} Y_t(w) &= Y_t(\omega,\omega') \\ &= X_t(\omega) \end{aligned} \tag{47}$$

defines obviously again a $\{W,(\Gamma_t),L\}$-process, where L is defined from K through (43). On the other hand, given a $\{\Omega',(\Pi'_t),K'\}$-process X', set

$$\begin{aligned} Y_t(w) &= Y_t(\omega,\omega') \\ &= X'_t(\omega') \end{aligned} \tag{48}$$

Then, (48) defines clearely a $\{W,(\Gamma_t),L\}$-process, where L is related to K' by (45). To summarize:

THEOREM 5: connections preserve causal processes.

5. CONCLUSION AND POSSIBLE EXTENSIONS.

We have presented a model to handle the causality in synchronous real time systems. The major feature of this model is that it takes into account the fact that real time synchronous systems are mainly governed by external input stimuli. As a consequence, events as well as processes were considered *as functions of the input stimuli*. This allowed us to introduce the notions of *clocks* and *causal processes*, to distinguish events and processes which satisfy a reasonable time correctness property. A complete set of primitive operators were defined to build any clock from the original one (i.e. from the clock of the input stimuli), and the change of time references was analysed.

5.1. On the use of this model for the design of the language SIGNAL.

This model to handle time in synchronous real time systems and languages was at the origin of the language SIGNAL (Le Guernic & al.(1984,1985)). The consequences of this point of view were the following.

5.1.1. Data flows and events are functions of the input stimuli.

A program (or "process" in the framework of the above mentioned references) is a closed black-box whose communication links with the external world are fixed and known in a static way. Of major importance are the *inputs* which represent the possible external stimuli. Such stimuli are modeled in the present work by the symbol "ω". In the language SIGNAL, data flows and events are thus considered as *functions of the input stimuli*, i.e. of ω. As a consequence, all the static timing verifications are performed on functions; see Le Guernic & al.(1985) for the notion of "clock calculus", a tool which is not mentioned here, but which is a key tool to build explicitely connections such as (35).

Note that a similar attempt is made in Caspi & Halbwachs (1985) to get

static verification of the correctness of precedence in real time systems, a different problem we do not investigated in this paper. The present approach, however, could serve as a basis to extend systematically the above mentioned work to handle the same kind of problems when the set of input stimuli plays a major role.

5.1.2. Filtering and multiplexing are primitives of the language SIGNAL.

Roughly speaking, a SIGNAL program specifies an oriented network; data flow along the paths of this network, and are transformed at the nodes (nodes are referred to as "black boxes"). The inputs of the program specify the input stimuli we mentioned above, i.e. the ω's, and the time reference, which is nothing but the ordering of these input data. New clocks are created by *temporal generators*, i.e. specific nodes which perform a temporal transformation on the data. Let us indicate how the filtering and multiplexing are effectively realized in SIGNAL.

The filtering.

This primitive actor takes as inputs: 1) a flow x, 2) a flow y; the values of x are delivered at the output port of this generator only when x and y are simultaneously available. This corresponds to the construction of the lemma 2.

The multiplexing.

This generator takes as input an ordered set $X=\{x^1, \ldots, x^n\}$ of causal processes with possibly different clocks. At a given time "t", some of the components of X_t only are available, say $\{x_t^{k_1}, \ldots, x_t^{k_m}\}$ where m≤n. Since the x_k's are causal processes, the index $m=m_t$ is causal as well. Then, this generator delivers at its output the ordered values

$$x_{(t,1)}, x_{(t,2)}, \ldots, x_{(t,m_t)}$$

which corresponds exactly to the way we built oversampled clocks in (21).

Clock synchronization.

Causal processes subject to different clocks can be used simultaneously (i.e. as if they had the same clock) provided their corresponding clocks are *synchronizable* in the sense of the definition 5. Explicit *synchro* statements are provided in the language SIGNAL for this purpose, since the static verification of the synchronizability property is a very difficult task (a mixture of logic and arithmetic verifications).

Causality and time correctness are guaranteed in the language SIGNAL.

Since the only temporal primitives of the language SIGNAL are the filtering, the multiplexing, and the delays (delays must be non negative!), the paragraphs 3 and 4 of this paper show that any SIGNAL program, whose syntax is correct and whose set of clocks is well defined and consistent (see Le Guernic & al. (1985) for the notion of "clock calculus"), satifies the causality and time correctness constraints.

5.2. Further work.

Several extensions of the present work are possible. First, there is no major difficulty to extend the present techniques to the continuous time **R** instead of the discrete time we used. This would allow to take into account the non determinism *at the input stimuli* but not the non determinism inside the real time system itself. This latter problem would require different techniques to handle at least the time uncertainty inside the real time system, see for example Cardelli (1982).

Second, this framework allows to extends some of the tools developped to analyse events to take into account that they are in fact functions or the input stimuli in real time systems. A first attempt in this direction is presented in Caspi & Halbwachs (1985) for the static verification of the correctness of the

time dependencies. A second example is the clock calculus developped for the language SIGNAL to verify the correctness of the constructed interconnections.

Bibliographical notes.

As it was indicated before, the notions we presented here are in fact directly borrowed from probability theory. First, histories are strongly connected to *filtered spaces*, i.e. spaces endowed with an increasingg family of σ-algebras; in fact, σ-algebras can be replaced by partitions when the underlying space enjoy certain properties of topological nature (e.g. so-called Lusin, Suslin, or Blackwell spaces. see Dellacherie & Meyer (1976)), thus resulting in an object very close to our histories. Second, our clocks are nothing but increasing families of *stopping times*, a basic notion in probability theory. Finally, our multiplexing construct is very similar to *stacks* or *gadgets* introduced in Ornstein's entropy theory of dynamical systems. What is indeed non classical in probability theory is our study of time changes; this is not surprising, since this study is entirely motivated by the requirement of programming modularity, an issue which has nothing to do with probability theory.

REFERENCES

Benveniste (1974) : A. Benveniste, "Processus stationnaires et mesures de Palm du flot special sous une fonction", these d'Etat univ. Paris VI, Seminaire de probabilites IX, Lect. notes in Math. vol 465.

Berry & al (1983) : G. Berry, S. Moisan, J.P. Rigault,"Towards a synchronous and semantically sound high level language for real time applications", *rep. centre de Math. Appl. Sophia Antipolis.*

Berry & Cosserat (1984) : G. Berry, L. Cosserat,"The ESTEREL Synchronous Programming Language and its mathematical Semantics", *Rapport de recherche No 327, INRIA*, Sophia-Antipolis, Septembre 1984.

Brookes & al.(1984) : S.D.Brookes, C.A.R.Hoare, A.W.Roscoe, "A Theory of Communicating Sequential Processes", JACM, Vol 31 No 3, 560-599.

Cardelli (1982) : L. Cardelli, "An algebraic approach to Hardware description and verification", thesis, Univ. of Edimburgh.

Caspi & Halbwachs (1982) : P. Caspi, N. Halbwachs,"Algebra of events : A model for parallel and real time systems," *Proc. of the 1982 int. conf. on parallel processing*, 1982, pp 150-159

Caspi & Halbwachs (1985) ; P. Caspi, N. Halbwachs, "Conception certifiee de systemes distribues: un exemple", rapport IMAG, to appear TSI.

Dellacherie & Meyer (1976) : C. Dellacherie, P.A. Meyer "*Probabilites et potentiel*" , 2nd edition, Hermann, Paris.

Kahn (1974) : G. Kahn,"The semantics of a simple language for parallel programming", *Information Processing 74*, North-Holland Publ. Comp. ,1974, pp 471,475.

Le Guernic & al.(1984) : P. Le Guernic, A. Benveniste, P. Bournai, T. Gautier, "SIGNAL: a data flow oriented language for signal processing", 1984-IEEE conf. on VLSI signal processing, Los Angeles, Nov. 1984.

Le Guernic & al. (1985) : P. Le Guernic, A. Benveniste, P. Bournai, T. Gautier, " SIGNAL: a data flow oriented language for signal processing", IRISA report nr.246.

Young (1982) : S.J. Young, "Real time languages: design and development", *Elis Horwood publishers*, 1982.